InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

LOTTO INU

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Lotto Inu
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0x48a0c51dF87d375AaF483281357Fa4a132ad4584
Blockchain	Binance Smart Chain
Centralization	Ownership Renounced
Commit	05306c72f85a8717397117adedd360000d018562
Website	https://lottoinu.com/
Report Date	April 17, 2024

ℹ️ Verify the authenticity of this report on our website: https://www.github.com/interfinetwork

EXECUTIVE SUMMARY

InterFi has performed the automated and manual analysis of solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical 🔴	Major 🟠	Medium 🟡	Minor 🟢	Unknown 🟤
Open	0	1	0	4	1
Acknowledged	0	0	0	1	1
Resolved	2	1	2	0	0
Major 🟠 Function	`burn()`				

ℹ️ Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

ℹ️ Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

SCOPE OF WORK

InterFi was consulted by Lotto Inu to conduct the smart contract audit of their solidity source codes. The audit scope of work is strictly limited to mentioned solidity file(s) only:

- LOTTOINU.sol

If source codes are not deployed on the main net, they can be modified or altered before main-net deployment. Verify the contract's deployment status below:

Public Contract Link	
https://bscscan.com/address/0x48a0c51df87d375aaf483281357fa4a132ad4584#code	
Contract Name	LOTTOINU
Compiler Version	0.8.6
License	MIT

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	○ Token Supply Manipulation ○ Access Control and Authorization ○ Assets Manipulation ○ Ownership Control ○ Liquidity Access ○ Stop and Pause Trading ○ Ownable Library Verification

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to solidity codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ️ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical 🔴	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 🟠	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 🟡	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- Privileged roles can be granted the power to `pause()` the contract in case of an external attack.
- Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- The client can lower centralization-related risks by implementing below mentioned practices:
- Privileged role's private key must be carefully secured to avoid any potential hack.
- Privileged role should be shared by multi-signature (multi-sig) wallets.
- Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.
- Renouncing the contract ownership, and privileged roles.
- Remove functions with elevated centralization risk.

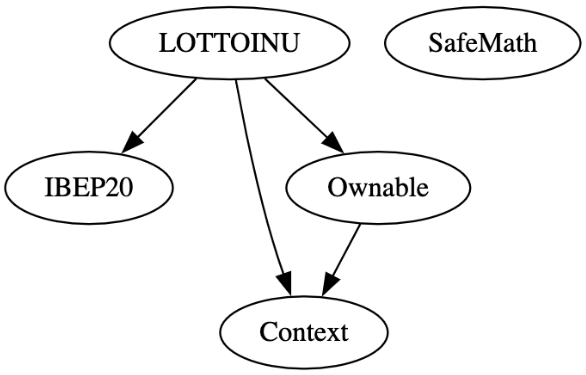
ℹ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💵	Function is payable
🔒	Function is internal
🔐	Function is private
❗	Function is important

```
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **Context** | Implementation |  |||
| └ | <Constructor> | Internal 🔒 | 🛑  | |
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **SafeMath** | Library |  |||
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Internal 🔒 | 🛑  | |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **LOTTOINU** | Implementation | Context, IBEP20, Ownable |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | mint | Public ❗ | 🛑  | onlyOwner |
| └ | burn | Public ❗ | 🛑  |NO❗ |
| └ | removeAddresses | Public ❗ | 🛑  |NO❗ |
| └ | selectWinner | Public ❗ |   |NO❗ |
| └ | getRandomNumber | Public ❗ |   |NO❗ |
| └ | removeAddressFromArray | Internal 🔒 | 🛑  | |
| └ | trackTransfer | Public ❗ | 🛑  |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _mint | Internal 🔒 | 🛑  | |
| └ | _burn | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _burnFrom | Internal 🔒 | 🛑  | |
```

INHERITANCE GRAPH



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major ●
CEN-10	Privileged role can mint tokens and add it to total supply	

Important `onlyOwner` centralized privileges are listed below:

```
renounceOwnership()
transferOwnership()
mint()
```

RECOMMENDATION

Deployers', owners', creators', and all other privileged roles' private-keys/access-keys/admin-keys should be secured carefully. These entities can have a single point of failure that compromises the security of the project.

Implement multi-signature wallets: Require multiple signatures from different parties to execute certain sensitive functions within contracts. This spreads control and reduces the risk of a single party having complete authority.

Use a decentralized governance model: Implement a governance model that enables token holders or other stakeholders to participate in decision-making processes. This can include voting on contract upgrades, parameter changes, or any other critical decisions that impact the contract's functioning.

RESOLUTION

Lotto Inu team has argued that privileged roles are used as intended, and smart contract ownership is renounced.

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor ●

All of the initially minted assets are sent to the project owner when deploying the contract. This can be an issue as the project owner can distribute tokens without consulting the community.

```
_totalSupply = 10 * 10**7; // 100 million tickets
_balances[msg.sender] = _totalSupply;
creator = msg.sender;//! first to run the smart contract is "creator"
emit Transfer(address(0), msg.sender, _totalSupply);
```

RECOMMENDATION

Project must communicate with stakeholders and obtain the community consensus while distributing assets.

ACKNOWLEDGEMENT

Lotto Inu team will distribute assets as per their pre-determined tokenomics.

Identifier	Definition	Severity
LOG-01	Lack of appropriate maximum boundary	Medium 🟡

Below mentioned function is set without any maximum input boundary:

```
mint()
```

RECOMMENDATION

Set maximum numerical mint cap to allow supply increase within reasonable limit.

RESOLUTION

Lotto Inu team has set maximum mint cap to allow supply increase within reasonable limit.

Identifier	Definition	Severity
LOG-02	Potential front-running	Minor ●

A malicious entity can manipulate the operation by front-running a transaction to purchase assets and make profits by back-running a transaction to sell assets. Users can see pending transactions and their details, hence, they can potentially front-run a transaction.

Below mentioned front-running scenarios may occur in this contract:

- Token Transfers and Lottery Entries:

The attacker may watch the pending transactions for high-value transfers or significant lottery actions. They may attempt to influence the lottery by entering it right before a high-value transfer they know is about to settle.

- Winner Selection:

When transaction for selecting winner is visible in mempool, the attacker may try to participate or withdraw from the lottery just before transaction is processed.

RECOMMENDATION

Implement commit-reveal scheme to obscure the transaction data until after transactions that can be influenced are finalized.

Identifier	Definition	Severity
LOG-03	Re-entrancy	Critical 🔴

Re-entrancy attacks occur when the external contract is called from within function, and this external contract then calls back into original function, possibly before first execution has fully completed.

Below mentioned function is vulnerable to re-entrancy attacks:

`_transfer()`

- Re-entrancy

While `_transfer()` function itself doesn't directly make external calls to other contracts that could execute untrusted code, it calls several internal functions which modify the state, such as, `trackTransfer()` and `selectWinner()` functions. Re-entrancy can occur in unusual ways; hence it is recommended to add re-entrancy guard.

- Checks-Effects-Interactions (CEI)

In `_transfer()` function, adjusting `totalTransfers` and then potentially distributing prize to randomly selected winner is violating the recommended CEI pattern. If winner's address is a contract, it may trigger logic that may re-enter `_transfer()` function

RECOMMENDATION

Use Checks Effects Interactions pattern when handing over the flow to an external entity and guard functions against re-entrancy attacks.

RESOLUTION

Lotto Inu team has applied `nonReentrant` modifier to `_transfer()` function.

Identifier	Definition	Severity
LOG-04	Complex state management	Minor 🟢

Lottery mechanism relies heavily on adjusting balances dynamically based on the total number of transfers. Use transfer as standalone function for token transfer, and handle lottery mechanism and payouts separately.

RECOMMENDATION

Detach lottery mechanism from token transfer process to simplify state management and reduce risk of bugs or attacks.

Identifier	Definition	Severity
LOG-05	Use of block properties for source of randomness	Unknown ●

Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp of block or `blockhash` to seed a random number, the miner can post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances.

Do not rely on `block.timestamp`, `blockhash` as a source of randomness.

Both the timestamp and the block hash can be influenced by miners to some degree. For example, malicious miners can run a casino payout function on a chosen hash and just retry a different hash if they did not receive any money.

RECOMMENDATION

Use a secure random number generator, such as Chainlink VRF (Verifiable Random Function) or decentralized oracles for random number generation.

Identifier	Definition	Severity
COD-06	Code improvement	Major ●

Burn operation should revert with a clear error message if the conditions are not met, rather than silently doing nothing.

```
function _burn(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: burn from the zero address");
    uint256 adjustedBalance = _balances[account] - ((_totalTransfers_ -
addressIndices[account]) * 1);
    require(adjustedBalance >= amount, "BEP20: burn amount exceeds balance");
    _balances[account] = _balances[account].sub(amount, "BEP20: burn amount exceeds
balance");
    _totalSupply = _totalSupply.sub(amount);
    emit Transfer(account, address(0), amount);
}
```

RECOMMENDATION

Incorporate recommended suggestion to improve burn function.

Identifier	Definition	Severity
COD-07	Potential denial-of-service (DoS)	Medium 🟡

In `selectWinner()` function, if array of entrants becomes large, the cost of generating a random number and selecting a winner may lead to high gas fees. Additionally, using delete on array elements in `removeAddressFromArray()` function can leave gaps, leading to probable issues in random selection logic. Avoid leaving gaps in arrays when removing entrants.

RECOMMENDATION

Use swap-and-delete pattern to maintain a compact array which can help reduce gas costs and avoid potential issues with gaps.

RESOLUTION

Lotto Inu team has updated logic in `removeAddressFromArray()` function.

Identifier	Definition	Severity
COD-08	Inappropriate function access modifiers	Critical 🔴

Below mentioned function should be `internal`:

`_transfer()`

RECOMMENDATION

Use appropriate access modifier to restrict access control.

RESOLUTION

Lotto Inu team has added `internal` modifier to `_transfer()` function.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown ●
COD-11	Block dependence for randomness generation	

Smart contract is interacting with third party protocols e.g., Dex Routers, External Contracts, Web 3 Applications, Open Zeppelin libraries. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, all of them can be compromised, and exploited. Moreover, upgrades in these entities can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect third party dependencies regularly, and mitigate severe impacts whenever necessary.

ACKNOWLEDGEMENT

Lotto Inu team claimed to inspect third party dependencies regularly, and push updates whenever required.

Identifier	Definition	Severity
COD-12	Lack of event-driven architecture	Minor 🟢

Certain significant state changes and operations (like adding/removing entrants or updating lottery details) do not emit events. This can make tracking the state of the contract difficult from an off-chain perspective.

RECOMMENDATION

Use events to track state changes. Events improve transparency and provide a more granular view of contract activity.

Identifier	Definition	Severity
VOL-01	Irrelevant code	Minor ●

Redundant code in `SafeMath`

RECOMMENDATION

When using Solidity 0.8.6, contract may not need `SafeMath` for basic arithmetic operations, as the compiler handles these checks automatically. Simplify the code by removing `SafeMath`.

DISCLAIMERS

InterFi Network provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, INTERFI NETWORK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT’S OR ANY OTHER INDIVIDUAL’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. InterFi Network does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than InterFi Network. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that InterFi Network is not responsible for the content or operation of such websites and social accounts and that InterFi Network shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT INTERFI NETWORK

InterFi Network provides intelligent blockchain solutions. We provide solidity development, testing, and auditing services. We have developed 150+ solidity codes, audited 1000+ smart contracts, and analyzed 500,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, etc.

InterFi Network is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 6+ casual contributors.

Website: https://interfi.network

Email: hello@interfi.network

GitHub: https://github.com/interfinetwork

Telegram (Engineering): https://t.me/interfiaudits

Telegram (Onboarding): https://t.me/interfisupport